Arrays

- Arrays are variables that allow storing multiple data that are related with each other.
- Arrays, similar to single value variables, can be of type int, char, double, etc.
- Let us represent temperature measurements taken every hour in a day:
  - $T_0$, $T_1$, ..., $T_{23}$
  - Array representation $T[0]$, $T[1]$, ..., $T[23]$
  - The numbers in brackets are called subscripts
  - The array has 24 elements, i.e. the array size is 24 ($N = 24$)
  - $T_i$ where $i$ goes from 0 to $N - 1$

Declaring Arrays

- Like any other variable, arrays are declared in the beginning of the function they are used

  - `type array_name[array_size];`
    - `type` is the type of array elements (int, double, char, ...)
    - `array_name` follows the same naming rules for the variables
    - `array_size` is an integer showing the number of elements in that array

  - `double T[24];`
Using Arrays

double x[8];

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0</td>
<td>2.1</td>
<td>5.2</td>
<td>9.3</td>
<td>2.5</td>
<td>5.5</td>
<td>1.9</td>
<td>-4.3</td>
</tr>
</tbody>
</table>

New x[3] = 7.5

i = 4; printf("%.1f", x[i]);  
Prints 2.5

printf("%.1f + %.1f = %.1f", x[0], x[6], x[0]+x[6]);  
Prints 10.0 + 1.9 = 11.9

i = 5; x[i] = x[i-1];  

i = 2; printf("%.1f", x[--i]);  
Prints 2.1. New i is 1.

i = 2; printf("%.1f", x[--i]);  
Prints 5.2. New i is 1.

Example

/*
 * This program allows the user to enter hourly temperature measurements for a day
 * then calculates minimum, maximum, and average temperature for that day
 */

#include <stdio.h>

int main(void)
{
    double T[24];  /* Temperature array */
    int i;  /* Array index */
    double Total = 0;  /* The sum of all temperatures */
    double Average;  /* The average temperature */
    double Minimum;  /* The minimum temperature */
    double Maximum;  /* The maximum temperature */

Example

/* Read hourly Temperature */
for(i=0; i<24; i++)
{
    printf("Please enter Temperature reading %2d> ", i+1);
    scanf("%lf", &T[i]);
    printf("T[%d] = %f\n", i, T[i]);
}

/* Calculate Average */
for(i=0; i<24; i++) Total += T[i];
Average = Total / 24;

/* Find minimum */
Minimum = T[0];
for(i=1; i<24; i++) if(T[i] < Minimum) Minimum = T[i];

/* Find maximum */
Maximum = T[0];
for(i=1; i<24; i++) if(T[i] > Maximum) Maximum = T[i];

/* Print Av, Min, and Max */
printf("The Average Temperature is:%-4.1f\n", Average);
printf("The Minimum Temperature is:%-4.1f\n", Minimum);
printf("The Maximum Temperature is:%-4.1f\n", Maximum);

Functions with Output Arguments

- Function with one input (non-array) and one output (as return)
  - Function Prototype
    - int ex1(int z);
  - Function Call
    - int x=0, y;
    - y = ex1(x);

- Function with one (non-array) input and one output (non-array)
  - Function Prototype
    - void ex1(int p, int *q);
  - Function Call
    - int x=0, y;
    - ex1(x, &y);

Array Elements as Arguments

- Function with one input (as array element) and one output (as return)
  - Function Prototype
    - int ex1(int z);
  - Function Call
    - int x[5], y;
    - x[3] = 0;
    - y = ex1(x[3]);

- Function with one input (non-array) and one output (as array element)
  - Function Prototype
    - void ex1(int p, int *q);
  - Function Call
    - int x=0, y[6];
    - ex1(x, &y[2]);
Arrays as Arguments

- Function with one input (as array) and one output (as return)
  - Function Prototype
    - `int ex1(int z[]):`
  - Function Call
    - `int x[5], i;
      for(i=0; i<5; i++) x[i] = i;
      y = ex1(x);`

- Function with one input (non-array) and one output (as array)
  - Function Prototype
    - `void ex1(int p, int q[]):`
  - Function Call
    - `int x=0, y[6];
      ex1(x, y);`

- Function with one input (array) and two outputs (one array one return)
  - Function Prototype
    - `int ex1(int p[], int q[]):`
  - Function Call
    - `int x[5], y[6], z;
      for(i=0; i<5; i++) x[i] = i;
      z = ex1(x, y);`
Example with Functions

/*
 * This program allows the user to enter hourly temperature measurements for a day
 * then calculates minimum, maximum, and average temperature for that day
 */

#include <stdio.h>
#define NUM_TEMP 24

void
ReadTemp(double x[], int N);
/* Reads N Temperature values to array */
double
CalcAve(double x[], double *Ave, int N);
/* Calculates the average of N temperatures */
double
FindMin(double x[], double *Min, int N);
/* Finds min of N temperatures */
double
FindMax(double x[], double *Max, int N);
/* Finds max of N temperatures */

int
main(void)
{

double T[NUM_TEMP];
/* Temperature array */
double Average;
/* The average temperature */
double Minimum;
/* The minimum temperature */
double Maximum;
/* The maximum temperature */

ReadTemp(T, NUM_TEMP);
/* Send the array to be filled and size of array */

CalcAve(T, &Average, NUM_TEMP);
/* T is the same as &T[0], it is an address */

FindMin(T, &Minimum, NUM_TEMP);
/* T is the same as &T[0], it is an address */

FindMax(T, &Maximum, NUM_TEMP);
/* T is the same as &T[0], it is an address */

printf("The Average Temperature is:%4.1f
n", Average);
printf("The Minimum Temperature is:%4.1f
n", Minimum);
printf("The Maximum Temperature is:%4.1f
n", Maximum);
}
Example with Functions

/* Reads Temperature values to array */
void ReadTemp(double x[], int N) /* x[] is the same as *x, it is a pointer */
{
    int i;
    for(i=0; i<N; i++)
    {
        printf("Please enter Temperature reading %2d> ", i+1);
        scanf("%lf", &x[i]);
    }
}

/* Calculates the average temperature */
void CalcAve(double x[], double *Ave, int N) /* x[] is the same as *x, it is a pointer */
{
    int i;
    double Total = 0;
    for(i=0; i<N; i++) Total += x[i];
    *Ave = Total / 24;
}

/* Finds min temperature */
void FindMin(double x[], double *Min, int N) /* x[] is the same as *x, it is a pointer */
{
    int i;
    *Min = x[0];
    for(i=1; i<N; i++) if(x[i] < *Min) *Min = x[i];
}

/* Finds max temperature */
void FindMax(double x[], double *Max, int N) /* x[] is the same as *x, it is a pointer */
{
    int i;
    *Max = x[0];
    for(i=1; i<N; i++) if(x[i] > *Max) *Max = x[i];
}
Value – Address – Pointer

Main Function

<table>
<thead>
<tr>
<th>Name</th>
<th>Addr.</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>T[0]</td>
<td>7000</td>
<td>0</td>
</tr>
<tr>
<td>T[1]</td>
<td>7001</td>
<td>2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>T[23]</td>
<td>7023</td>
<td>46</td>
</tr>
</tbody>
</table>

ReadTemp(T, 24);

Value

T[0] = 0
T[1] = 2

Address

&T[0] = 7000
&T[1] = 7001

ReadTemp Function

<table>
<thead>
<tr>
<th>Name</th>
<th>Addr.</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>x[ ]</td>
<td>7900</td>
<td>7000</td>
</tr>
</tbody>
</table>

ReadTemp(double x[ ], int N);

x[ ] = *(x = *(7000))
x[0] = *(x+0) = *(7000) = 0
x[1] = *(x+1) = *(7001) = 2

If Function not to modify Array

If we do not want the called function to modify an array that is passed to it as argument, put the reserved word const in front of the argument in the formal parameter list.

**Prototype:**

```c
void FindMin(const double x[ ], double *Min, int N);
```

**Call:**

```c
FindMin(T, &Minimum, 24);
```

**Definition:**

```c
void FindMin (const double x[ ], double *Min, int N) {
    int i;
    *Min = x[0];
    for(i=1;i<N; i++) if(x[i] < *Min) *Min = x[i];
}
```